

Introduction to Lists in Java

CPSC 2150

Arrays vs Lists

- So far in Java, we've just been using Arrays
- Arrays require that we know the size when we write the code
 - This is inflexible
 - If we need more entries, we have to declare a larger array then copy the old values over
- Lists are more flexible
 - We can easily add and remove elements to our list
 - We don't need to know the size

Lists in Java

- Later Lecture: The *Java Collections Framework (JCF)* is a group of interfaces and classes
 - See Java libraries package `java.util`
- **This Lecture:** A preview of `List` collection
 - One of the many collections in `java.util`
- We will talk about `Lists` and how exactly they work much later on
- For now, we'll just focus on how to use `Lists` in our program

Declaring a List

- Lists are a Generic Collection
 - We'll talk about this more later
 - For now, that just means that a List can hold any data type
 - As long as it's not a primitive data type
 - This is why Integer and other wrapper classes are so helpful
 - We have to tell the compiler what data type it will store
 - We use the <> angle brackets for this
 - Data type is denoted with <T>
- Declare and initialize a List of Strings
 - `List<String> myStringList = new ArrayList<String>();`
- Declare and initialize a List of Pencils
 - `List<Pencil> myPencilList = new ArrayList<Pencil>();`

List vs ArrayList?

- Our declared type (`List`) does not match the type of the constructor in Initialization (`ArrayList`)
- How can this happen?
 - This is because of **Interfaces**, which we'll talk about in a later lecture
 - For now, just go with the flow

Adding items to our List

- `add(<T>)` method
 - Add our object of type `T` to the end of the `List`
 - `List` must also be of type `T`
 - `myStringList.add("Hello");`
- `add(int pos, <T>)` method
 - Add our object of type `T` to the `List` in position `pos`
 - `List` must also be of type `T`
 - `myStringList.add(0, "Sam");`

Removing Items from our List

- `remove(int pos)` method
 - Removes the object at position `pos` from our list
- `removeRange(int fromIndex, int toIndex)` method
 - `fromIndex` – index of first element to be removed
 - `toIndex` – index after last element to be removed
 - Removes the range of objects in the list

List get and set

- Allows you to access a specific index in our `List` (like an Array)
 - **Note:** not as time efficient as it is with an array
- `get(int pos)` method
 - Returns the object in position `pos` in the list
- `set(int pos, <T> x)` method
 - Change the object in the list at position `pos` to be `x`

Memory Management Methods

- `size()` method
 - Returns the number of elements in the `List`
- `isEmpty()` method
 - Returns *true* if the `List` currently has no elements

contains method

- The `contains(<T> x)` method will tell us if the `List` has `x` in it.
 - Still has to search through the list, which can take time
- In order to do this, it will call the `.equals(x)` on every position of the list
 - If we are using a user defined class, we need to override `.equals()` for this to work
 - This is an example of why every class inherits from `Object`. That means every class has an `equals` method, even if it just checks the reference value.
 - This means we can safely have a `contains` method, and other similar methods

for loop

- We can use our special for loop syntax with `Lists` as well
- Makes it easier than using the methods to get to a particular position.
- ```
for (<T> varName: listName) {
 // refer to the variable with varName
}
```

# Using List (and ArrayList)

```
import java.util.List;
import java.util.ArrayList;

List<String> list = new ArrayList<String>();
list.add("Hello");
list.add("there");
list.add(0, "Sam");
System.out.println(list.get(1)); // "Hello"

for (String str : list) {
 System.out.println(str);
} // prints "SamHellothere"
```

# Lists

- This is just a basic introduction
- We will discuss Lists in more detail as we learn about interfaces, generics, and collections
- This will be enough for you to work with `Lists`
- You will need this in labs and in future homework